

# PATENT ABSTRACTS OF JAPAN

(11)Publication number : 2000-149008

(43)Date of publication of application : 30.05.2000

(51)Int.Cl. G06T 3/40  
H04N 1/387  
H04N 1/409

(21)Application number : 11-243399

(71)Applicant : SEIKO EPSON CORP

(22)Date of filing : 30.08.1999

(72)Inventor : ISHIDA MASANORI

(30)Priority

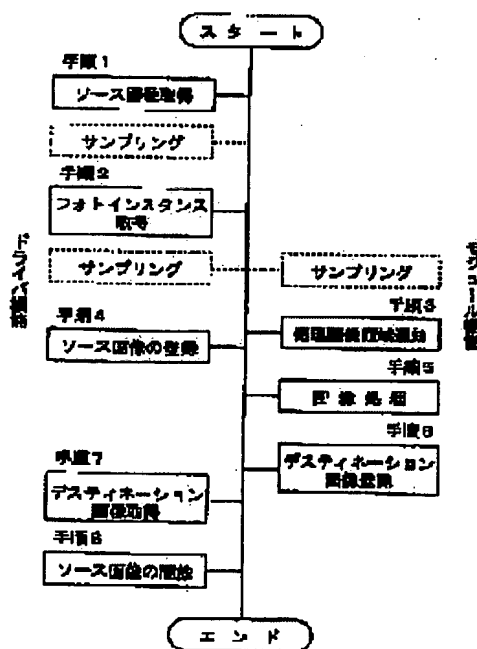
Priority number : 10245337 Priority date : 31.08.1998 Priority country : JP

## (54) MEDIUM RECORDING IMAGE PROCESSING PROGRAM AND IMAGE PROCESSING DEVICE AND METHOD

(57)Abstract:

**PROBLEM TO BE SOLVED:** To provide a medium which records an image processing program that can prevent cases, where the image processing cannot be executed according to a higher level of image processing technique and also to provide the image processing device and method.

**SOLUTION:** A module notifies the driver of a processing image area (procedure 3) after a source image is acquired (procedure 1) and after a photo instance acquisition processing (procedure 2). At the same time, the necessary fringing areas are added with every image processing module, so that the source image (S-unit) is registered by the driver after the necessary fringing area is added to a requested processing image area (procedure 4). Based on a registered S-unit, image processings are successively executed by utilizing the enhancement and enlargement processings (procedure 5). Thus, it is possible to secure and process extremely simply a necessary area in an, even when the image processing requires a fringing area.



## LEGAL STATUS

[Date of request for examination]

11.04.2000

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number] 3467727

[Date of registration] 05.09.2003

[Number of appeal against examiner's decision of rejection]

[Date of requesting appeal against examiner's decision of rejection]

[Date of extinction of right]

Copyright (C); 1998,2003 Japan Patent Office

(19) 日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11) 特許出願公開番号

特開2000-149008

(P2000-149008A)

(43) 公開日 平成12年5月30日 (2000.5.30)

| (51) Int.Cl. <sup>7</sup> | 識別記号 | F I           | テーマコード (参考) |
|---------------------------|------|---------------|-------------|
| G 0 6 T 3/40              |      | G 0 6 F 15/68 | 3 5 5 C     |
| H 0 4 N 1/387             |      | H 0 4 N 1/387 |             |
| 1/409                     |      | 1/40          | 1 0 1 D     |

審査請求 未請求 請求項の数 7 O L (全 20 頁)

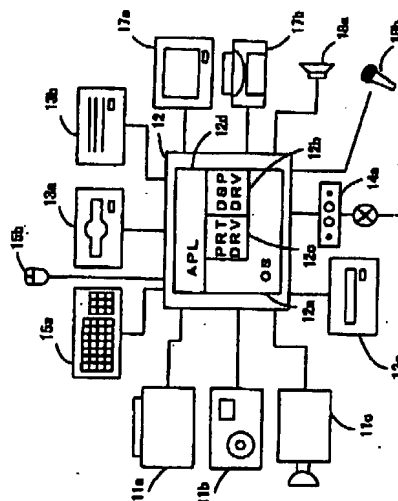
|              |                        |          |   |
|--------------|------------------------|----------|---|
| (21) 出願番号    | 特願平11-243399           | (71) 出願人 | 000002369<br>セイコーエプソン株式会社<br>東京都新宿区西新宿2丁目4番1号 |
| (22) 出願日     | 平成11年8月30日 (1999.8.30) | (72) 発明者 | 石田 正紀<br>長野県諏訪市大和3丁目3番5号 セイコーエプソン株式会社内        |
| (31) 優先権主張番号 | 特願平10-245337           | (74) 代理人 | 100096703<br>弁理士 横井 俊之                        |
| (32) 優先日     | 平成10年8月31日 (1998.8.31) |          |   |
| (33) 優先権主張国  | 日本 (J P)               |          |   |

(54) 【発明の名称】 画像処理プログラムを記録した媒体、画像処理装置および画像処理方法

## (57) 【要約】

【課題】 既存の画素についての処理を実行するためにはその周縁の画素についての情報を取り入れておかなければならない場合もあり、そのような場合だと画像領域の周縁部分で必要な画素の情報を参照できなくなるという課題があった。

【解決手段】 ソース画像の取得処理後 (手順1)、フォトインスタンス取得処理 (手順2) を経たら、モジュールの側からドライバの側に向けて処理画像領域を通知するが (手順3)、このときに各画像処理モジュールで必要な周縁領域を加えていくことにより、要求された処理画像領域に必要な周縁領域を加えた上でソース画像 (S-Unit) をドライバの側が登録し (手順4)、これを利用して順次画像処理 (強調処理と拡大処理 (手順5)) を実行することになり、画像処理が周縁領域を必要とする場合にも極めてシンプルに必要な領域を確保して処理を実行することができる。



## 【特許請求の範囲】

【請求項1】 ドットマトリクス状に配置された画素からなる画像を表す領域にオブジェクト画像を適宜配置するにあたり、同オブジェクト画像を表現するために複数の分割オブジェクトデータが生成され、各オブジェクト画像ごとに所定の画像処理をコンピュータに実行させる画像データ補間処理プログラムを記録した媒体であって、

同画像処理プログラムは、上記オブジェクト画像の分割オブジェクトデータが入力されるドライバと、各オブジェクト画像ごとに画像処理を実行するモジュールとを具備し、かつ、上記ドライバは上記分割オブジェクトデータをスプールするとともに、上記モジュールが処理対象とする画像領域のオブジェクトデータをスプールファイルからメモリ上に展開して利用可能とする際、モジュールが処理対象とする画像領域に加えて各画像処理において参照する周縁領域を加えて上記メモリ上に展開することを特徴とする画像処理プログラムを記録した媒体。

【請求項2】 上記請求項1に記載の画像処理プログラムを記録した媒体において、上記モジュールが個別の画像処理を実行する複数の画像処理オブジェクトを有する場合に、各画像処理オブジェクトの画像処理において必要とする周縁領域を予め加えておいた画像領域を上記ドライバがメモリ上に展開することを特徴とする画像処理プログラムを記録した媒体。

【請求項3】 上記請求項2に記載の画像処理プログラムを記録した媒体において、上記モジュールの各画像処理オブジェクトは順次自己の画像処理において必要とする周縁領域を処理対象とする画像領域に加えて残りの画像処理オブジェクトに通知していき、最後の画像処理オブジェクトにおいて周縁領域を加えられた画像領域を上記ドライバがメモリ上に展開することを特徴とする画像処理プログラムを記録した媒体。

【請求項4】 上記請求項1～請求項3のいずれかに記載の画像処理プログラムを記録した媒体において、上記ドライバは、周縁領域を加えた画像領域を含めた上で分割オブジェクトデータ単位でメモリ上に展開することを特徴とする画像処理プログラムを記録した媒体。

【請求項5】 上記請求項1～請求項4のいずれかに記載の画像処理プログラムを記録した媒体において、上記ドライバは、メモリ上に展開した分割オブジェクトデータのうちで利用頻度の高いものを限定して利用可能にすることを特徴とする画像処理プログラムを記録した媒体。

【請求項6】 ドットマトリクス状に配置された画素からなる画像を表す領域にオブジェクト画像を適宜配置するにあたり、同オブジェクト画像を表現するために複数の分割オブジェクトデータが生成され、各オブジェクト画像ごとに所定の画像処理する画像処理装置であって、上記オブジェクト画像の分割オブジェクトデータが入力

されるデータ管理手段と、

各オブジェクト画像ごとに画像処理を実行するデータ処理手段とを具備し、

かつ、上記データ管理手段は上記分割オブジェクトデータをスプールするとともに、上記データ処理手段が処理対象とする画像領域のオブジェクトデータをスプールファイルからメモリ上に展開して利用可能とする際、同データ処理手段が処理対象とする画像領域に加えて各画像処理において参照する周縁領域を加えて上記メモリ上に展開することを特徴とする画像処理装置。

【請求項7】 ドットマトリクス状に配置された画素からなる画像を表す領域にオブジェクト画像を適宜配置するにあたり、同オブジェクト画像を表現するために複数の分割オブジェクトデータが生成され、各オブジェクト画像ごとに所定の画像処理する画像処理方法であって、上記オブジェクト画像の分割オブジェクトデータが入力されるデータ管理工程と、

各オブジェクト画像ごとに画像処理を実行するデータ処理工程とを具備し、

上記データ管理工程では、上記分割オブジェクトデータをスプールするとともに、上記データ処理工程で処理対象とする画像領域のオブジェクトデータをスプールファイルからメモリ上に展開して利用可能とする際、同データ処理工程で処理対象とする画像領域に加えて各画像処理において参照する周縁領域を加えて上記メモリ上に展開することを特徴とする画像処理方法。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】本発明は、画像処理プログラムを記録した媒体、画像処理装置および画像処理方法に関する。

【0002】

【従来の技術】コンピュータなどでは、画像をドットマトリクス状に配置された画素で表しているが、近年、仮想的なドットマトリクス状の領域に対して、オブジェクト画像を適宜配置して一枚の画像を表す手法が利用されている。すなわち、文字だけを表すオブジェクト画像や、イメージを表すオブジェクト画像を個別に用意し、所定の順番で重ね合わせて画像を完成させる。このような画像を扱う際、コンピュータの処理の都合上、一つのオブジェクト画像を複数の分割にすることが多い。例えば、オブジェクト画像が大きくなるにつれてデータ量も極めて多大となり、一体として処理するには処理資源の負担が大きくなるからである。

【0003】一方、コンピュータで複数の段階に分けて画像を処理することも可能であり、アプリケーションのように内部において自由に処理できるものの他、オペレーティングシステムの一部として機能するドライバやモジュールによってメモリなどの制約を強く受けながら処理しなければならないものもある。すなわち、分割され

たオブジェクトデータを連結して元のオブジェクト画像を生成することなく、画像処理をしなければならないものがある。従来、モジュールがオブジェクトデータの処理を実行する場合にドライバに対してオブジェクトデータへのアクセスを要求するが、それはモジュールで処理しようとしている画像領域に過ぎなかった。

【0004】

【発明が解決しようとする課題】上述した従来のものにおいては、オブジェクトデータがモジュールに提供するのは本来の処理対象である画像領域に対応したものであった。しかし、画像処理が既存の単独画素だけに基いて処理を実行しているときには問題が生じないものの、既存の画素についての処理を実行するためにはその周縁の画素についての情報を取り入れておかなければならない場合もあり、そのような場合だと画像領域の周縁部分で必要な画素の情報を参照できなくなるという課題があった。

【0005】本発明は、上記課題にかんがみてなされたもので、画像処理の高度化に伴って画像処理を実行できなくするのを防止することが可能な画像処理プログラムを記録した媒体、画像処理装置および画像処理方法の提供を目的とする。

【0006】

【課題を解決するための手段】上記目的を達成するため、請求項1にかかる発明は、ドットマトリクス状に配置された画素からなる画像を表す領域にオブジェクト画像を適宜配置するにあたり、同オブジェクト画像を表現するために複数の分割オブジェクトデータが生成され、各オブジェクト画像ごとに所定の画像処理を実行する画像処理プログラムを記録した媒体であって、同画像処理プログラムは、上記オブジェクト画像の分割オブジェクトデータが入力されるドライバと、各オブジェクト画像ごとに画像処理を実行するモジュールとを具備し、かつ、上記ドライバは上記分割オブジェクトデータをスプールするとともに、上記モジュールが処理対象とする画像領域のオブジェクトデータをスプールファイルからメモリ上に展開して利用可能とする際、モジュールが処理対象とする画像領域に加えて各画像処理において参照する周縁領域を加えて上記メモリ上に展開する構成としてある。

【0007】上記のように構成した請求項1にかかる発明においては、ドライバにオブジェクト画像の分割オブジェクトデータが入力されると、モジュールが各オブジェクト画像ごとに画像処理を実行するが、ここで上記ドライバは上記分割オブジェクトデータをスプールするとともに、上記モジュールが処理対象とする画像領域のオブジェクトデータをスプールファイルからメモリ上に展開して利用可能としている。そして、この際、同ドライバは同モジュールが処理対象とする画像領域に加えて各画像処理において参照する周縁領域を加えて上記メモリ

上に展開している。すなわち、本画像処理プログラムでは、ドットマトリクス状に配置された画素からなる画像を表す領域にオブジェクト画像を適宜配置するにあたり、同オブジェクト画像を表現するために複数の分割オブジェクトデータが生成され、各オブジェクト画像ごとに所定の画像処理を実行するが、処理の都合上、ドライバは分割オブジェクトデータをスプールし、モジュールが処理対象とする画像領域のオブジェクトデータをスプールファイルからメモリ上に展開しており、このときにメモリ上に展開される領域は本来の処理対象とする画像領域だけでなく、モジュールで実行する画像処理において参照する周縁領域まで加えてある。

【0008】ところで、モジュールは必ずしも単独の画像処理を実行するものに限られる必要はなく、複数の画像処理を実行できるものであっても構わない。このような場合に好適な一例として、請求項2にかかる発明は、請求項1に記載の画像処理プログラムを記録した媒体において、上記モジュールが個別の画像処理を実行する複数の画像処理オブジェクトを有する場合に、各画像処理オブジェクトの画像処理において必要とする周縁領域を予め加えておいた画像領域を上記ドライバがメモリ上に展開する構成としてある。上記のように構成した請求項2にかかる発明においては、複数の画像処理オブジェクトにおけるそれぞれの画像処理において必要とする周縁領域を予め加えておき、上記ドライバはかかる周縁領域を含めた画像領域をメモリ上に展開している。

【0009】このように各画像処理オブジェクトの画像処理において必要とする周縁領域を予め加えておくにあたり、請求項3にかかる発明は、請求項2に記載の画像処理プログラムを記録した媒体において、上記モジュールの各画像処理オブジェクトは順次自己の画像処理において必要とする周縁領域を処理対象とする画像領域に加えて残りの画像処理オブジェクトに通知していき、最後の画像処理オブジェクトにおいて周縁領域を加えられた画像領域を上記ドライバがメモリ上に展開する構成としてある。上記のように構成した請求項3にかかる発明においては、各画像処理オブジェクトが自己の画像処理において必要とする周縁領域を処理対象とする画像領域に加えることができ、複数の画像処理オブジェクトがある場合に、これを数珠つなぎとして、それぞれの画像処理オブジェクトが次段の画像処理オブジェクトに対して周縁領域を加えた画像領域を通知していく。これにより、最後の画像処理オブジェクトにおいて周縁領域を加えられた画像領域を参照すれば、全ての画像処理オブジェクトにおいて必要となる周縁領域が加えられた画像領域が分かり、ドライバはかかる最終の画像領域をメモリに展開すればよい。

【0010】さらに、請求項4にかかる発明は、請求項1～請求項3のいずれかに記載の画像処理プログラムを記録した媒体において、上記ドライバは、周縁領域を加

えた画像領域を含めた上で分割オブジェクトデータ単位でメモリ上に展開する構成としてある。メモリ上に展開されるオブジェクトデータは必ずしも処理対象となる画像領域そのままである必要はなく、余裕を持って展開されても構わない。すなわち、参照される周縁領域も含めてメモリに展開されていなければ困るが、逆に余分に展開されていることは支障がない。このため、上記のように構成した請求項 4 にかかる発明においては、周縁領域を加えた画像領域を含めた上でドライバが分割オブジェクトデータ単位でメモリ上に展開する。

【0011】分割オブジェクトデータの数極めて多くなると、処理の低下を免れなくなるが、そのような場合に好適な一例として、請求項 5 にかかる発明は、請求項 1～請求項 4 のいずれかに記載の画像処理プログラムを記録した媒体において、上記ドライバは、メモリ上に展開した分割オブジェクトデータのうちで利用頻度の高いものを限定して利用可能にする構成としてある。メモリ上に展開した分割オブジェクトデータは必ずしも全てを必要としないこともあるため、上記のように構成した請求項 5 にかかる発明においては、利用頻度の高い分割オブジェクトデータを限定して利用可能にすることによって不要な分割オブジェクトデータを処理対象とすることなく、処理速度などの低下を防止する。

【0012】以上においては、発明の思想の具現化例として、ソフトウェアを記録した記録媒体について説明した。むしろ、その記録媒体は、磁気記録媒体であってもよいし光磁気記録媒体であってもよいし、今後開発されるいかなる記録媒体においても全く同様に考えることができる。また、一次複製品、二次複製品などの複製段階については全く問う余地無く同等である。その他、供給方法として通信回線を利用して行なう場合でも本発明が利用されていることにはかわりない。さらに、一部がソフトウェアであって、一部がハードウェアで実現されている場合においても発明の思想において全く異なるものではなく、一部を記録媒体上に記憶しておいて必要に応じて適宜読み込まれるような形態のものとしてあってもよい。

【0013】しかしながら、本発明にかかるソフトウェアとしてのみ利用されるというわけでないことは明らかであり、その一例として、請求項 6 にかかる発明は、ドットマトリクス状に配置された画素からなる画像を表す領域にオブジェクト画像を適宜配置するにあたり、同オブジェクト画像を表現するために複数の分割オブジェクトデータが生成され、各オブジェクト画像ごとに所定の画像処理する画像処理装置であって、上記オブジェクト画像の分割オブジェクトデータが入力されるデータ管理手段と、各オブジェクト画像ごとに画像処理を実行するデータ処理手段とを具備し、かつ、上記データ管理手段は上記分割オブジェクトデータをスプールするとともに、上記データ処理手段が処理対象とする画像領域のオ

ブジェクトデータをスプールファイルからメモリ上に展開して利用可能とする際、同データ処理手段が処理対象とする画像領域に加えて各画像処理において参照する周縁領域を加えて上記メモリ上に展開する構成としてある。上記のように構成した請求項 6 にかかる発明における動作は、ソフトウェアの場合と同様である。

【0014】また、このような手法は必ずしも実体のある装置に限られる必要はなく、その方法としても機能することは容易に理解できる。このため、請求項 7 にかかる発明は、ドットマトリクス状に配置された画素からなる画像を表す領域にオブジェクト画像を適宜配置するにあたり、同オブジェクト画像を表現するために複数の分割オブジェクトデータが生成され、各オブジェクト画像ごとに所定の画像処理する画像処理方法であって、上記オブジェクト画像の分割オブジェクトデータが入力されるデータ管理工程と、各オブジェクト画像ごとに画像処理を実行するデータ処理工程とを具備し、上記データ管理工程では、上記分割オブジェクトデータをスプールするとともに、上記データ処理工程で処理対象とする画像領域のオブジェクトデータをスプールファイルからメモリ上に展開して利用可能とする際、同データ処理工程で処理対象とする画像領域に加えて各画像処理において参照する周縁領域を加えて上記メモリ上に展開する構成としてある。

【0015】

【発明の効果】以上説明したように本発明は、画像処理の高度化に伴って処理対象の画像領域の周縁領域まで参照する必要が生じる場合にも画像処理を実行することが可能な画像処理プログラムを記録した媒体を提供することができる。また、請求項 2 にかかる発明によれば、複数の画像処理によって参照する周縁領域が異なる場合も画像処理を実行することができる。さらに、請求項 3 にかかる発明によれば、各画像処理オブジェクトが順次周縁領域を通知していくことにより、必要な周縁領域の管理を行う煩わしさを減少させることができる。さらに、請求項 4 にかかる発明によれば、ドライバはオブジェクトデータ単位でスプールしているため、さらに必要領域を考慮して切り出すということが必要でなくなり、処理速度は向上する。さらに、請求項 5 にかかる発明によれば、利用頻度の低い分割オブジェクトデータを処理対象としないことによって処理速度の低下を防止することができる。さらに、請求項 6 にかかる発明によれば、同様の効果を奏することが可能な画像処理装置を提供でき、請求項 7 にかかる発明によれば、画像処理方法を提供できる。

【0016】

【発明の実施の形態】以下、図面にもとづいて本発明の実施形態を説明する。図 1 は本発明の一実施形態にかかる画像処理プログラムを適用した画像処理システムを実現するハードウェアの一例としてのコンピュータシステ

ム10をブロック図により示している。本コンピュータシステム10は、画像データを直接的に入力する画像入力デバイスとして、スキャナ11aとデジタルスチルカメラ11bとビデオカメラ11cとを備えており、コンピュータ本体12に接続されている。それぞれの入力デバイスは画像をドットマトリクス状の画素で表現した画像データを生成してコンピュータ本体12に出力可能となっており、ここで同画像データはRGBの三原色においてそれぞれ256階調表示することにより、約1670万色を表現可能となっている。

【0017】コンピュータ本体12には、外部補助記憶装置としてのフロッピーディスクドライブ13aとハードディスク13bとCD-ROMドライブ13cとが接続されており、ハードディスク13bにはシステム関連の主要プログラムが記録されており、フロッピーディスクやCD-ROMなどから適宜必要なプログラムなどを読み込み可能となっている。また、コンピュータ本体12を外部のネットワークなどに接続するための通信デバイスとしてモデム14aが接続されており、外部のネットワークに同公衆通信回線を介して接続し、ソフトウェアやデータをダウンロードして導入可能となっている。この例ではモデム14aにて電話回線を介して外部にアクセスするようにしているが、LANアダプタを介してネットワークに対してアクセスする構成とすることも可能である。

【0018】ここで、外部補助記憶装置のうち、フロッピーディスクドライブ13aやCD-ROMドライブ13cについては、記録媒体自身が交換可能であり、この記録媒体に画像データが記録された状態で供給されることにより、画像入力デバイスの一手段ともなりうる。また、モデム14aやLANアダプタを介してネットワークにアクセスした場合、このネットワークから画像データが供給されることもあり、このような場合も画像入力デバイスの一手段となりうる。その他、コンピュータ本体12の操作にキーボード15aやポインティングデバイスとしてのマウス15bも接続され、さらに、マルチメディア対応のためにスピーカ18aやマイク18bを備えている。一方、画像出力デバイスとして、ディスプレイ17aとカラープリンタ17bとを備えている。ディスプレイ17aについては水平方向に800画素と垂直方向に600画素の表示エリアを備えており、各画素毎に上述した1670万色の表示が可能となっている。むろん、この解像度は一例に過ぎず、640×480画素であったり、1024×768画素であるなど、適宜、変更可能である。

【0019】また、印刷装置としてのカラープリンタ17bはインクジェットプリンタであり、CMYKの四色の色インクを用いて記録媒体たる印刷用紙上にドットを付して画像を印刷可能となっている。画像密度は360×360dpiや720×720dpiといった高密度

印刷が可能となっているが、階調表現については色インクを付すか否かといった2階調表現となっている。色インクについては、かかる四色のものに限らず、色の薄いライトシアンやライトマゼンタを加えた六色によってドットの目立ちを低減させることも可能であるしインクジェット方式に限らずカラートナーを利用した静電写真方式などを採用することも可能である。このような画像入力デバイスを使用して画像を入力しつつ、画像出力デバイスに表示あるいは出力するため、コンピュータ本体12内では所定のプログラムが実行されることになる。そのうち、基本プログラムとして稼働しているのはオペレーティングシステム(OS)12aであり、このオペレーティングシステム12aにはディスプレイ17aでの表示を行わせるディスプレイドライバ(DSPDRV)12bとカラープリンタ17bに印刷出力を行わせるプリンタドライバ(PRT DRV)12cが組み込まれている。これらのドライバ12b、12cの類はディスプレイ17aやカラープリンタ17bの機種に依存しており、それぞれの機種に応じてオペレーティングシステム12aに対して追加変更可能である。また、機種に依存して標準処理以上の付加機能を実現することもできるようになっている。すなわち、オペレーティングシステム12aという標準システム上で共通化した処理体系を維持しつつ、許容される範囲内での各種の追加的処理を実現できる。

【0020】この基本プログラムとしてのオペレーティングシステム12a上でアプリケーション12dが実行される。アプリケーション12dの処理内容は様々であり、操作デバイスとしてのキーボード15aやマウス15bの操作を監視し、操作された場合には各種の外部機器を適切に制御して対応する演算処理などを実行し、さらには、処理結果をディスプレイ17aに表示したり、カラープリンタ17bに出力したりすることになる。かかるコンピュータシステム10では、画像入力デバイスであるスキャナ11aなどで写真などを読み取って画像データを取得することができる他、デジタルスチルカメラ11bで撮影した画像データを取得したり、ビデオカメラ11cで撮影した動画としての画像データを取得することができる。このような画像データは最終的に画像出力デバイスとしてのディスプレイ17aで表示したり、カラープリンタ17bで印刷することになるが、元の画像データのままで写りが悪いなどの問題があることが多く、そのような場合には、何らかの修正が行われる。この修正を行うのは、一般的にはフォトタッチなどのアプリケーション12dなどであるが、本画像処理システムではアプリケーション12dが印刷処理を実行するときに出力するオブジェクトデータに対して、プリンタドライバ12cが同修正処理を実行する。かかる印刷処理に際しては、アプリケーション12dはオペレーティングシステム12aに対してオブジェクトデータを

出力することになる。

【0021】図2は、アプリケーション12dから出力されるオブジェクトデータである印刷イメージの処理単位の変化を示している。同図において、アプリケーション12dにて生成された印刷イメージPIは、オペレーティングシステム12aに出力される。そして、オペレーティングシステム12aは描画命令群12a1に基づいて、入力した印刷イメージPIを所定の描画命令に置き換えて印刷データを生成する。ここで、オペレーティングシステム12aが描画命令に置き換えた印刷データは、本来作画されたイメージをページ単位の座標をそのままにして形成されている印刷イメージPIを、処理の都合上、ランダムな矩形のオブジェクトデータに分割されている。そして、オペレーティングシステム12aは、この分割した矩形のオブジェクトデータをページ単位の座標とは無関係に、すなわち、印刷イメージPIの作画されたイメージに対応することなくランダムな位置に配置し、分割オブジェクトデータa(1)～a(n)を形成する。

【0022】すなわち、それぞれの分割オブジェクトデータa(1)～a(n)において矩形で区分された領域の内部データは、座標が連続している領域に対応したデータを表わしているものの、相互に隣り合う分割オブジェクトデータa(1)～a(n)は座標に関しては何等連続性を有するものではない。従って、隣り合う分割オブジェクトデータa(1)～a(n)は不連続なデータとなっており、この形成された分割オブジェクトデータa(1)～a(n)の並び順に基づいて印刷を行っても、正常な印刷イメージPIを再現することは不可能になっている。このように分割オブジェクトデータa(1)～a(n)に分割された印刷イメージPIについて印刷を行う場合、分割オブジェクトデータa(1)～a(n)は、プリンタドライバ12cに入力されるとともに、適宜色調処理や拡大・縮小処理などの画像処理が実行される。そして、これらの画像処理の後に、カラープリンタ17bが処理可能なラスタデータb(1)～b(n)を形成して、カラープリンタ17dに出力する。

【0023】かかる場合、プリンタドライバ12cは、上述したラスタデータb(1)～b(m)をバンド単位で順次形成するために、ランダムに配置された分割オブジェクトデータa(1)～a(n)から相互に関連のある分割オブジェクトデータa(1)～a(n)を選択しスプールファイル12c1に格納する。そして、このスプールファイル12c1に格納した分割オブジェクトデータa(1)～a(n)から所定のデータ領域を抽出してメモリ12c2上に展開し、上述した色調処理や拡大・縮小処理などを実行する同プリンタドライバ12cが備える画像処理を実行するモジュール12c3において利用可能にする。このメモリ12c2はハードウェア化

されたプリンタドライバ12cに配設されている所定の記憶領域であってもよいし、コンピュータ本体に取り付けられた図示しないEEPROMなどの記録領域であってもよい。

【0024】図2は、アプリケーション12dから出力されるオブジェクトデータである印刷イメージの処理単位の変化を示している。同図において、アプリケーション12dにて生成された印刷イメージPIは、オペレーティングシステム12aに出力される。そして、オペレーティングシステム12aは描画命令群12a1に基づいて、入力した印刷イメージPIを所定の描画命令に置き換えて印刷データを生成する。ここで、オペレーティングシステム12aが描画命令に置き換えた印刷データは、オペレーティングシステム12aでの処理の都合上、複数の矩形からなるオブジェクトデータに分割される。そして、オペレーティングシステム12aは、この分割した矩形のオブジェクトデータをページ単位の座標とは無関係に、すなわち、印刷イメージPIの作画されたイメージに対応することなくランダムな位置に配置し、分割オブジェクトデータa(1)～a(n)を形成する。

【0025】すなわち、それぞれの分割オブジェクトデータa(1)～a(n)において矩形で区分された領域の内部データは、作画座標が連続している領域に対応したデータを表わしているものの、相互に隣り合う分割オブジェクトデータa(1)～a(n)は座標に関しては連続性を有するものあるとは限られない。従って、隣り合う分割オブジェクトデータa(1)～a(n)は不連続なデータとなっており、この形成された分割オブジェクトデータa(1)～a(n)の並び順に基づいて印刷を行っても、正常な印刷イメージPIを再現することは不可能になっている。このように分割オブジェクトデータa(1)～a(n)に分割された印刷イメージPIについて印刷を行う場合、所定の分割オブジェクトデータa(1)～a(n)がプリンタドライバ12cに入力されるとともに、適宜色調処理や拡大・縮小処理などの画像処理が実行される。そして、これらの画像処理の後に、カラープリンタ17bが処理可能なラスタデータb(1)～b(n)を形成して、カラープリンタ17dに出力する。

【0026】かかる場合、プリンタドライバ12cは、上述したラスタデータb(1)～b(m)をバンド単位で順次形成するために、ランダムに配置された分割オブジェクトデータa(1)～a(n)から相互に関連のある分割オブジェクトデータa(1)～a(n)を選択してスプールファイル12c1に格納する。そして、このスプールファイル12c1に格納した分割オブジェクトデータa(1)～a(n)から上述した色調処理や拡大・縮小処理などの画像処理を実行する同プリンタドライバ12cの備えるモジュール12c3が画像処理の処理



11

対象とするデータ領域を抽出し、同モジュール12c3が利用可能な状態でメモリ12c2上に展開する。

【0027】次に、プリンタドライバ12cがオペレーティングシステム12aに組み込まれた状態でのシステム上の概略構成を図3に示す。同図において、アプリケーション12dから印刷要求があると、API層から本プリンタドライバ12cが提供する機能管理オブジェクトP1にアクセスされる。この機能管理オブジェクトP1は、分割画像管理オブジェクトP2によって上記分割オブジェクトデータa(1)～a(n)の管理を実行しつつ、処理画像管理オブジェクトP3によって処理画像を取得する。この処理画像を取得するにあたって機能管理オブジェクトP1は同処理画像管理オブジェクトP3に対して必要領域を要求するし、その処理の前提として予め分割オブジェクトデータa(1)～a(n)に関して分割画像登録の処理を実行する。

【0028】すなわち、処理画像管理オブジェクトP3が実際の画像処理を実行する拡大処理オブジェクトP4と強調処理オブジェクトP5を介して分割画像管理オブジェクトP2に対して処理対象となる画像領域を通知するため、分割画像管理オブジェクトP2は既に登録されている分割オブジェクトデータa(1)～a(n)のうちの必要なものだけをメモリ12c2上に展開して利用可能とし、展開された結果を利用して強調処理オブジェクトP5と拡大処理オブジェクトP4が画像処理を実行することになる。従って、本発明にかかる画像処理プログラムにおいて、上述した機能管理オブジェクトP1と、分割画像管理オブジェクトP2と、処理画像管理オブジェクトP3とが本発明にかかるドライバを構成するとともに、拡大処理オブジェクトP4と、強調処理オブジェクトP5とが本発明にかかる画像処理を実行するモジュールを構成する。

【0029】図4は入力される分割オブジェクトデータと取得される処理画像とを示している。同図において、元の画像をソース(Source)画像と呼ぶとともに、拡大処理と強調処理を実施された画像をデスティネーション(Destination)画像と呼ぶことにすると、入力される分割オブジェクトデータは9分割されたS-Unit、a～iである。ここで、デスティネーション画像は、一体として処理することも可能であるが、この場合も分割して処理するものとし、各分割画像をD-Unit、A～Cと呼ぶ。D-Unit、A～Cを得るためには処理前の画像領域を利用可能であることが必要であるが、入力されているのはS-Unit、a～iであって、全てを利用可能となっているか否かは分からない。従って、処理画像管理オブジェクトP3から分割画像管理オブジェクトP2に対して対応するS-Unit、a～iを利用可能とするように要求し、そのようにして利用可能にされるのがS-Area、a～cである。ただし、このS-Area、a～cは領域として

12

は拡大前のD-Unit、A～Cと一致するものではなく、周縁領域を含むものとなっている。このように周縁領域を含むのは、拡大処理オブジェクトP4と強調処理オブジェクトP5で画像処理する際に処理対象の画素だけでは処理が不可能であり、その周囲の画素も参照しなければならないからである。

【0030】次に、各画像処理で参照する周縁領域を図5および図6に示す。具体的な処理内容については詳述しないが、図5に示す強調処理では対象画素を基準として上方と左方に1画素、右方と下方に2画素を参照する必要がある。また、図6に示す拡大処理では、対象画素を基準として右方左方と下方に1画素、上方に2画素を参照する必要がある。周縁領域を参照する必要がある具体的な演算例として、フィルタ処理が上げられる。この場合の強調処理と拡大処理の例にあてはめれば、強調処理においては4×4画素分のフィルタを利用し、拡大処理においては3×4画素分のフィルタを利用するので、周縁領域が必要になるといえる。図7は処理画像管理オブジェクトP3から分割画像管理オブジェクトP2に必要領域が通知される過程を示している。同図において、最終的に取得される処理画像の領域が、最初の必要領域S-Area'として処理画像管理オブジェクトP3に要求されると、同処理画像管理オブジェクトP3は、拡大処理オブジェクトP4にこれを通知する。すると、拡大処理オブジェクトP4では自己の拡大倍率が2倍であるときには上記必要領域S-Area'の元となる画像領域は1/2倍である。ただし、参照する画素は上方、右方、下方、左方に対してそれぞれ1、2、2、1画素ずつであることを強調処理オブジェクトP5に通知する。この強調処理オブジェクトP5では拡大を行わないため、本来的に必要なのは拡大処理オブジェクトP4で必要とする画像領域であり、上記必要領域S-Area'の1/2倍であるとともに、その周縁領域の(1, 2, 2, 1)画素である。しかし、自己においても周縁領域を参照する必要があり、さらに周縁領域の(2, 1, 1, 1)画素を加えて分割画像管理オブジェクトP2に通知する。すなわち、このような修正を加えた画像領域S-Areaは必要領域S-Area'の1/2倍の領域と、その周縁領域の(3, 3, 3, 2)画素を加えた領域となる。むろん、このような周縁領域の大きさは拡大処理オブジェクトP4や強調処理オブジェクトP5の演算手法に依存しており、他の演算例では強調処理において上下左右の全てに2画素を要求しつつ、拡大処理で上方と左方に1画素、右方と下方に2画素を要求するというようなものもある。

【0031】一方、分割画像管理オブジェクトP2は、機能管理オブジェクトP1を介して個々の分割オブジェクトデータが入力されており、領域としての情報を管理しながら実体についてはスプールファイルに格納して分割画像登録を実施している。スプールファイルはハード

ディスク13b上に格納されるため、他の処理オブジェクトは直接的に参照したり利用することはできない。しかしながら、この分割画像管理オブジェクトP2は以上のようにして通知された画像領域S-Areaを他のオブジェクトから利用できるようにするため、メモリ上に展開する。この際、必要とされた画像領域S-Areaのそのままをメモリ上に展開するのではなく、図8に示すように、あくまでも画像領域S-Areaを包含するに足るS-Unit. a~iがどれにあたるかを調査し、該当する全てのS-Unit. a~iをメモリ上に展開する。ここで分割オブジェクトデータの数が少なければ画像領域S-Areaを包含するS-Unit. a~iがどれであるかを判断するのは容易である。各S-Unit. a~iの実体データは、ハードディスク13b上に格納されるものの、各領域に関するヘッダ情報は、図9に示すようなテーブルに格納されている。各テーブルには次のS-Unit. a~iの情報開始位置を示すポインタnext\_ptrが備えられている。本来であれば、このポインタをたどることによって全てのS-Unit. a~iのヘッダ情報を参照することができるが、各S-Unit. a~iが画像領域S-Areaの一部とでも重なり合うか否かを判断するのはいくつかの比較演算を繰り返さなければならず、演算回数が多大となってしまう。一方、分割オブジェクトユニットの配置情報などから必ずしも全てのS-Unit. a~iと対比する必要もなく、現実的には参照すべきS-Unit. a~iは限られている。例えば、図9に示す例であれば、S-Areaを展開するに際し、斜線を示すS-Unit. a~iだけを参照できればよく、一時的にはあるがこのテーブルにおけるポインタnext\_ptrを振り替えておくようにしている。

【0032】本実施形態においては、ポインタnext\_ptrにより参照するS-Unit. a~iを振り替えて間接的にS-Areaを形成するS-Unit. a~iを指定し、S-Areaを高速に生成可能な構成を採用しているが、むしろ、S-Areaを高速に生成可能にする手法は限定されるものではない。図21に示すようにS-Areaの生成に必要なS-Unit. a~iをハードディスクと比較して相対的に高速アクセス可能な主メモリやキャッシュメモリなどのメモリに格納させ、S-Areaの生成時に参照させるようにしてもよいし、図22に示すように各S-Unit. a~iにID1~9を振り分けるとともに、このID1~9により順不同に配置されたS-Unit. a~iがソース画像の元画像を形成できるような順番に配置するIDテーブルを形成し、S-Areaの生成時に参照するようにしてもよい。

【0033】以上のようにして分割画像管理オブジェクトP2が画像領域S-Areaを包含する全てのS-Unit. a~iをメモリ上に展開したら、強調処理オブ

ジェクトP5が強調処理を実行し、次いで拡大処理オブジェクトP4が拡大処理を実行する。強調処理では対象画素の周縁領域として(2, 1, 1, 1)画素が必要であり、強調処理の結果として生成される画像領域は図10に示すように周縁領域として(2, 1, 1, 1)画素を省いた破線内の領域となる。また、同様にして拡大処理では対象画素の周縁領域として(1, 2, 2, 1)画素が必要であり、強調処理の結果として生成される画像領域は図11に示すように周縁領域として(1, 2, 2, 1)画素を省いた破線内の領域となる。むしろ、拡大処理は内部に格子点を補間生成するものであるから、得られる処理画像は縦横2倍に拡大される結果、最初の必要領域S-Area'と一致することになる。

【0034】次に、分割画像管理オブジェクトP2がメモリ上にS-Areaを展開する他の実施例を説明する。かかる実施例におけるプリンタドライバ12cがオペレーティングシステム12aに組み込まれた状態でのシステム上の概略構成を図23に示す。同図において、API層から本プリンタドライバ12cが提供する機能管理オブジェクトP100にアクセスされ、この機能管理オブジェクトP100は、分割画像管理オブジェクトP200によって分割オブジェクトデータの管理を実行しつつ、処理画像管理オブジェクトP300より拡大および色調に関する画像処理が実施された画像データを1ラインごとと取得し、取得したライン単位の画像データを積み上げてデスティネーション画像を形成する画像データを生成する。

【0035】具体的には、各モジュールにおいて色調処理オブジェクトP500が分割画像管理オブジェクトP200より引き渡された所定のライン単位の画像データに対して色調処理を実施し、所定のライン単位の画像データを形成し拡大処理オブジェクトP400に引き渡す。次に、拡大処理オブジェクトP400は、この色調処理が為された画像データに対して拡大処理を実施し、1ラインの画像データを形成し処理画像管理オブジェクトP400に引き渡し、機能管理オブジェクトP100は処理画像管理オブジェクトP300より1ラインごとに画像データを取得しつつ積み上げてデスティネーション画像を生成する。

【0036】ここで、本実施形態においては、拡大処理オブジェクトP400は、1ラインの拡大処理後の画像データを生成するため、同1ラインと参照ラインとを含め5ラインの画像データを必要とする。従って、色調処理オブジェクトP500に対して5ラインの画像データの引き渡しを通知することになる。また、色調処理オブジェクトP500は、1ラインの色調処理後の画像データを生成するため、当該1ラインの画像データのみでよい。従って、分割画像管理オブジェクトP200に対して1ラインの画像データの引き渡しを要求することになる。むしろ、これらの処理にて必要とするライン数は、

特に限定されるものではなく、各オブジェクトの処理内容に応じて適宜変更可能である。

【0037】次に、図24に入力される分割オブジェクトデータと取得される処理画像とを示す。同図において、元の画像をソース(Source)画像と呼ぶとともに、拡大処理と色調処理を実施された処理画像をデスティネーション(Destination)画像と呼ぶことにする。また、入力される分割オブジェクトデータは9分割されたS-Unit、A~Iである。ここで、デスティネーション画像は、一体として処理することも可能であるが、本実施例においてはライン単位に分割してラインごとに生成され、このラインごとに生成されたものを積み上げて最終的なデスティネーション画像が生成されるものとする。また、デスティネーション画像を分割するライン単位の各分割画像をD-ライン、0~nと呼ぶ。ここで、nは整数であり、生成されるデスティネーション画像の総ライン数を示している。最初に、D-ライン、0を得るために、処理画像管理オブジェクトP300は、拡大処理オブジェクトP400に対して1ラインの引き渡しを通知する。ここで、拡大処理オブジェクトP400は、1ラインの拡大処理を実行するために、上述したように5ラインの画像データを必要とする。従って、次の色調処理オブジェクトP500に対して5ラインの引き渡しを通知する。色調処理オブジェクトP500は、色調処理を実行するにあたり1ラインの画像データにて処理可能であるため、分割画像管理オブジェクトP200に1ラインの画像データの引き渡しを通知する。

【0038】そして、分割画像管理オブジェクトP200は、分割画像データS-Unit、A~Iより所定の1ラインの画像データを形成し、色調処理オブジェクトP500に引き渡す。色調処理オブジェクトP500は、この引き渡された1ラインのデータに色調処理を実施し、色調処理が終わると、拡大処理オブジェクトP400からは5ラインを要求されているため、5ライン分の色調処理が終わるまで、分割画像管理オブジェクトP200に1ラインを引き渡しを通知し、分割画像管理オブジェクトは、要求に従い順次1ラインを生成し、色調処理オブジェクトに渡す。そして、色調処理オブジェクトP500は、引き渡された1ラインの画像データに順次色調処理を実行する。色調処理オブジェクトP500は、5ラインに対する色調処理が終わると、拡大処理オブジェクトP400にこの5ラインを引き渡す。拡大処理オブジェクトP400は、この引き渡された5ラインの内後ろ4ラインを参照して先頭1ラインの強調処理を実施する。そして、色調処理も強調処理も実施された1ラインのデスティネーション画像が生成され、処理画像管理オブジェクトP300に引き渡される。

【0039】最初の1ラインのデスティネーション画像が生成されると、以降、順次nラインまで、1ラインご

とデスティネーション画像が生成される。この1ラインの画像を生成するにあたり、処理画像管理オブジェクトP300は、拡大処理オブジェクトP400に1ラインの引き渡しを通知する。ここで、拡大処理オブジェクトP400は、既に、色調処理が終わっている5ラインを引き渡されている。従って、この5ラインのうち先頭の1ラインを不参照の扱いとし、2番目のラインについて拡大処理を実施するために、色調処理オブジェクトP500に1ラインの引き渡しを通知する。この通知を受けると色調処理オブジェクトP500は、分割画像管理オブジェクトP200に1ラインの引き渡しを通知する。分割画像管理オブジェクトP200は、S-Unit、A~Iより所定の1ラインを形成し、色調処理オブジェクトP500に引き渡す。色調処理オブジェクトP500は、この引き渡された1ラインに対して色調処理を実施し、拡大処理オブジェクトP400に引き渡す。拡大処理オブジェクトP400は、この引き渡された1ラインを既に引き渡されている4ラインの最後尾に付加して、5ラインの処理領域を形成し、上述した2番目のラインについて拡大処理を実施する。この2番目の1ラインについて拡大処理が終わると、処理画像管理オブジェクトP300に引き渡す。機能管理オブジェクトP100は、このように処理画像管理オブジェクトP300が取得した各オブジェクトにて各処理が実施されたライン単位の画像データを順次積み上げてデスティネーション画像を生成する。

【0040】以上のようにして、最初に分割画像管理オブジェクトP200が色調処理オブジェクトP500に対して1ラインの画像領域をメモリ上に展開して引き渡し、色調処理オブジェクトP500が色調処理を実行し、色調処理オブジェクトP500において5ラインの画像データを生成すると、拡大処理オブジェクトP400に引き渡し、拡大処理を実行する。そして、5ラインより生成された1ラインの画像データを処理画像管理オブジェクトP300に引き渡す。そして、以降は、処理画像管理オブジェクトP300、拡大処理オブジェクトP400、色調処理オブジェクトP500、分割画像管理オブジェクトP200間にて1ラインの引き渡し通知および形成し、あるいは、画像処理を実行した1ラインの画像データを逆方向にて順次引き渡し、機能管理オブジェクトP100にて積み上げることによりデスティネーション画像を生成する。本実施形態においては、分割画像管理オブジェクトP200から色調処理オブジェクトP500に引き渡される1ラインの画像データおよび色調処理オブジェクトP500から拡大処理オブジェクトP400に引き渡される5ラインあるいは1ラインの画像データがS-Areaを構成することはいうまでもない。また、本実施形態においては、画像処理を実行する順番を色調処理オブジェクトP500→拡大処理オブジェクトP400としているが、むしろ、これに限定さ

れるものではなく、拡大処理オブジェクトP400→色調処理オブジェクトP500としてもよいし、所定の設定により、拡大処理オブジェクトP400あるいは色調処理オブジェクトP500のいずれか一方のみを実行するようにしてもよい。

【0041】ここで、図12にプリンタドライバ12cにおける手続の流れを示す。同図においては、オペレーティングシステム12aに組み込まれた状態でのプリンタドライバ12cの概略構成を示しているが、図3ではプリンタドライバ12c単独での視点で示しており、さらにプリンタドライバ12c内を本来のドライバの機能に近い処理と、付加的な機能に近い処理とに分割し、左欄にドライバ機能をまとめ、右欄にモジュール機能をまとめている。すなわち、手順1においてはドライバ機能としてソース画像の取得処理があり、これは機能管理オブジェクトP1と分割画像管理オブジェクトP2の機能といえる。次に、手順2はフォトインスタンス取得処理があるが、これは各オブジェクト画像ごとに別個・並行に処理するために現在の実行処理対象を確認するためのものである。

【0042】以上が前処理に対応し、処理の流れとしては手順3においてモジュールの側からドライバの側に向けて処理画像領域を上述したように通知する

一方、手順4では要求された処理画像領域を含むソース画像(S-Unit, a~i)をドライバ側が登録し、これに続いて手順5では画像処理として強調処理および拡大処理を実行することになる。なお、画像処理は、オブジェクト画像全体についての評価を行っておき、その評価結果に基づいて処理を実行する。例えば、あるオブジェクト画像が全体的に暗いのであれば、画像処理で明るく修整すればよい。この場合、オブジェクト画像は分割オブジェクトデータに分割されてしまっているから、それぞれのオブジェクトデータをまとめて評価するか、個別的にオブジェクトデータについて評価しつつも、最終的には各オブジェクト画像ごとに評価をまとめる必要がある。

【0043】本来、分割オブジェクトデータとオブジェクト画像との対応関係は付けられていないのが通常であり、この対応付けはプリンタドライバ12cの側で行わなければならない。この対応付けの一つの手法として、仮想領域に分割オブジェクトデータを展開してオブジェクト画像を連結する手法があるし、別の手法として分割オブジェクトデータ同士の重なり合いや隣接具合から同じオブジェクト画像であるか否かを判断する手法がある。いずれの手法を採用するかはシステムの相違によってまちまちである。

【0044】一方、上述したように画像処理のためにオブジェクト画像を評価するには、オブジェクト画像全体にわたって画素の情報をサンプリングすることが有効であり、簡易である。このようなサンプリングは、オブジ

ェクト画像と分割オブジェクトデータとの対応関係が分かる前後のどちらで実行しても良い。すなわち、対応関係が分かる前に分割オブジェクトデータごとにサンプリングしておき、対応関係が分かってからサンプリング結果を統合することもできるし、対応関係が分かってから各オブジェクト画像ごとに対応する分割オブジェクトデータをサンプリングすることもできる。このようにサンプリングの実行タイミングは条件に応じてさまざまであり、図12においても実行可能な三つのタイミングに対応して破線で示している。なお、ドライバの側で行うといってもサンプリング用のモジュールを用意することも可能である。

【0045】手順5の画像処理は、このサンプリング結果を利用して実行され、生成された画像がデスティネーション画像となって手順6にて登録される。そして、登録されたデスティネーション画像は手順7でドライバ側にて出力されることになる。そして、最終的にデスティネーション画像が取得されたら手順8でソース画像をメモリから解放する。ここでソース画像の解放タイミングは特に制限されるものではない。例えば、一つのS-Areaごとに登録・解放を実行するとすれば使用メモリ領域を低減できる反面、作業が煩わしい。これに対して使用メモリ領域の制限が少ない場合は、登録・解放の作業をこまめに行わなくても良いと言える。

【0046】なお、上述した例では、デスティネーション画像を取得する意味で必要領域を通知しており、生成されるデスティネーション画像は任意のメモリ領域に展開されればよい。これに対してスプールファイル自体に対して拡大処理を実行したい場合もある。この場合は、拡大処理された画像をスプールファイルに対して付加し、元画像ファイルのポインタを拡大処理された画像ファイルにリンクさせればよい。例えばスプールファイルから読み出して仮想画像を構築し、画像処理を施してから元のスプールファイルを書き換えたいという処理があるとする。かかる場合に、拡大処理をスプールファイル中の画像ファイルに施してしまえば仮想画像上で拡大処理を実現しなくても良くなる。しかしながら、拡大処理すればスプールファイル中の画像ファイルよりも大きくなるから、拡大後の画像ファイルで書き換えることはできない。従って、拡大後の画像データをスプールファイルの後に付加していき、ポインタのリンクを張り替えるようにするのである。

【0047】次に、拡大処理の具体的な手法について説明する。一般に、ドットマトリクス状の画素からなる画像についてその格子間に新たに画素を生成する処理を拡大処理と呼んで広く実行されている。これは、ディスプレイ17aの解像度とカラープリンタ17bの解像度に相違があるような場合、画素単位で割り当ててしまうと一定の大きさに表示されなくなるといった不具合を解消するためである。ここで、拡大処理には、次に示す手法が

10

20

30

40

50

知られている。

1. 最近隣内挿法(単純水増しコピー)

2. 3次たたみ込み補間(以下、3次補間という)

前者のものは数学的に最も高速な拡大手法であるし、後者のものはきれいな拡大結果を得られる反面で数学的には大量の演算を必要として時間がかかるという特徴を有している。いずれにしても、これらは画像処理の技術用語としてはフィルタ処理と呼ばれている。ここで、図13や図14は別々のフィルタ[1]、[2]を使用して拡大前の画像aから拡大後の画像Aを生成する原理を示している。この場合、拡大前の画像aの数点を利用してフィルタ[1]、[2]にかけることにより、拡大後の画像Aのある1点を求めることができる。従って、拡大後の画像Aの全画素について本来的には演算を繰り返すことになる。

【0048】ところで、フィルタ[1]、[2]には処理結果について差があるはずである。すなわち、前者は荒く見え、後者は滑らかに見えるように思われる。しかしながら、拡大結果を使用して印刷した場合、人間の目の認識の限界により必ずしもその差を認識することがで

$$P = [f(Y_0) f(Y_1) f(Y_2) f(Y_3)] \begin{pmatrix} P_{00} & P_{10} & P_{20} & P_{30} \\ P_{01} & P_{11} & P_{21} & P_{31} \\ P_{02} & P_{12} & P_{22} & P_{32} \\ P_{03} & P_{13} & P_{23} & P_{33} \end{pmatrix} \begin{pmatrix} f(X_0) \\ f(X_1) \\ f(X_2) \\ f(X_3) \end{pmatrix}$$

なお、関数  $f(t)$  については、

$$f(t) = \{\sin(\pi t)\} / \pi t$$

で表され、UまたはVをWに置き換えることにより、

$$t_0 = 1 + (W - |W|)$$

$$t_1 = (W - |W|)$$

$$t_2 = 1 - (W - |W|)$$

$$t_3 = 2 - (W - |W|)$$

で表すことができる。

【0050】このような3次補間演算は、一般的に多くの演算量を要することになるが、補間点の位置によっては高速化が可能であり、より具体的には整数倍の拡大に限れば処理量はかなり低減することができる。なお、ここでいう整数倍とは原点が格子線の交差点上に置かれる※

$$t_0 = 1, \quad t_1 = 0, \quad t_2 = 1, \quad t_3 = 2$$

となる。このとき、 $f(t)$  は、上式より以下のようになる。

$$f(0) = 1, \quad f(1) = 0, \quad f(2) = 0$$

よって、Y軸に関しては格子線上に存在する場合のX軸上の補間点は、

【数2】

$$P = [P_{01} \ P_{11} \ P_{21} \ P_{31}] \begin{pmatrix} f(X_0) \\ f(X_1) \\ f(X_2) \\ f(X_3) \end{pmatrix}$$

※きない。すなわち、高度な演算を要するフィルタ[2]だけで拡大した場合と、フィルタ[1]、[2]を併用して拡大した場合とにおいて、拡大結果の差を認識できない場合がある。図15は、後者の手順を示しており、元の画像aに対してフィルタ[2]を利用して拡大処理を実行し、一度、画像a'を生成した後に、今度はフィルタ[1]を利用して拡大処理を実行し画像Aを得ている。むろん、先のフィルタ[2]の拡大処理は3次補間を要するのでモジュールで実行し、フィルタ[1]の拡大処理はドライバで実現するようにしてもよい。この場合、もちろんフィルタ[1]とフィルタ[2]の拡大倍率によっては肉眼でも差が分かるが、そのような差が分からない範囲の組合せであるならば、処理時間のかかるフィルタ[2]にて生成する画像a'の面積は小さいため高速な処理を期待できる。なぜなら、拡大処理後の画像の面積に比例して演算量も多くなるからである。

【0049】図16は、3次補間の場合の既存の格子点と補間点との配置を示すとともに、数1は、この3次補間のフィルタ処理を行列演算式で示している。

【数1】

$$P = [f(Y_0) f(Y_1) f(Y_2) f(Y_3)] \begin{pmatrix} P_{00} & P_{10} & P_{20} & P_{30} \\ P_{01} & P_{11} & P_{21} & P_{31} \\ P_{02} & P_{12} & P_{22} & P_{32} \\ P_{03} & P_{13} & P_{23} & P_{33} \end{pmatrix} \begin{pmatrix} f(X_0) \\ f(X_1) \\ f(X_2) \\ f(X_3) \end{pmatrix}$$

※ようになる倍率のことであって、拡大処理前後の格子点数の比を意味するものではない。図17は、この定義でいうところの2倍の拡大であり、図18は、3倍の拡大である。図中、大きな白丸が原点で、小さい黒丸が補間点である。これらの図から分かるように、整数倍の拡大においては、補間点のいくつかは格子線上に配置される。以下、具体的な高速化手段について説明する。

【0051】(1) 格子線上の補間点の演算簡易化  
整数倍の拡大においては、補間点のいくつかは格子線上に配置される。従って、数1は、簡易化(演算量が減る)される。すなわち、補間点が格子線上に存在するということは、W(UあるいはVのいずれか)が0になることを意味する。よって、

$$t_0 = 1, \quad t_1 = 0, \quad t_2 = 1, \quad t_3 = 2$$

によって算出することになり、X軸に関しては格子線上に存在する場合のY軸上の補間点は、

【数3】

$$P = [f(Y_0) f(Y_1) f(Y_2) f(Y_3)] \begin{pmatrix} P_{10} \\ P_{11} \\ P_{12} \\ P_{13} \end{pmatrix}$$

によって算出することになる。この数2、数3から明らかのように拡大処理の演算を簡易化することが可能になる。

【0052】(2) 複数のフィルタを用意することによる高速化

整数倍のときに格子線上に位置する補間点はもっとも特異な例であるが、格子線上にない場合でも倍率を固定するとともに、補間点ごとのフィルタ処理を用意すれば、 $f(t)$  は定数化できるので演算処理を高速化できる。かかる場合、 $f(t)$  を定数化すると、

【数4】

$$P = [ABCD] \begin{pmatrix} P_{00} & P_{10} & P_{20} & P_{30} \\ P_{01} & P_{11} & P_{21} & P_{31} \\ P_{02} & P_{12} & P_{22} & P_{32} \\ P_{03} & P_{13} & P_{23} & P_{33} \end{pmatrix} \begin{pmatrix} K \\ L \\ M \\ N \end{pmatrix}$$

により表されることになる。ここにおいて、 $A \sim D$ 、 $K \sim N$  は、補間点の場所 $W$ によって一意的に決定されるものであり、予め演算してテーブルに保存しておけばよい。

【0053】ここで、拡大処理において2倍の場合について図17を参照してみると、局所的には4個の点を補間しなければならないので拡大処理前の画像 $a$ を画像 $A$ に拡大する場合、4つのフィルタを用意することになる。これを図で表したものが図19であり、複数のフィルタ処理を用意することを前提とすれば、 $f(t)$  の定数化が可能になる。 $f(t)$  の定数化が実現する具体的な高速化は以下のような理由による。まず、 $f(t)$  の定数化そのこと自体による高速化が可能である。なぜならば、 $f(t)$  の定数化は、 $f(t)$  で表せられる3次補間関数の演算の除去を意味するからである。次に、演算種別の変化によって高速化が可能であ

\* 30 る。一般に、CPUにとって同じ1回の掛け算でも、

“変数 \* 変数” より “定数 \* 変数” の方が高速な処理となる。従って、数1と数4とを比較した場合、数4の方が高速化されることが分かる。さらに、一方の定数化によって“定数 \* 変数” の掛け算を演算テーブルにて代用することにより、高速化することができる。

【0054】以上の高速化の効果をまとめると次のようになる。1つの補間点を求めるために、1つのフィルタ処理を行なう従来の手法によれば、

10 ・3次補間関数の演算

・20回の“変数 \* 変数”の演算

が必要である。しかしながら、上に述べたように複数のフィルタ処理を前提とした場合には、

・16回の演算テーブルへのアクセス

・4回の“定数 \* 変数”の演算

により実現できる。従って、複数のフィルタを用意した方が高速化を実現できる。また、先に述べた格子線上の補間点の演算簡略化についても、同様にして複数のフィルタ処理を用意して実現可能である。そして、このような複数のフィルタ処理を用意することによる高速化と、格子線上にある補間点の式の簡略化による高速化は、整数倍であれば何倍でも有効である。

【0055】(3) その他の高速化

(a) 2の $n$ 乗倍の拡大における特異な補間点の演算式の簡略化

2の $n$ 乗倍の拡大においては( $n$ は1以上の整数)、最近傍の原点と原点の2分の1にある場所の点を結んだ格子線(図17の破線)に補間点が置かれるが、 $X$ 軸上の場合は、

\* 30 【数5】

$X$ 軸上

$$P = [ABCD] \begin{pmatrix} P_{00} & P_{10} & P_{20} & P_{30} \\ P_{01} & P_{11} & P_{21} & P_{31} \\ P_{02} & P_{12} & P_{22} & P_{32} \\ P_{03} & P_{13} & P_{23} & P_{33} \end{pmatrix} \begin{pmatrix} K \\ L \\ L \\ K \end{pmatrix}$$

により表され、 $Y$ 軸上の場合は、

\* 30 【数6】

$Y$ 軸上

$$P = [ABBA] \begin{pmatrix} P_{00} & P_{10} & P_{20} & P_{30} \\ P_{01} & P_{11} & P_{21} & P_{31} \\ P_{02} & P_{12} & P_{22} & P_{32} \\ P_{03} & P_{13} & P_{23} & P_{33} \end{pmatrix} \begin{pmatrix} K \\ L \\ M \\ N \end{pmatrix}$$

により表されることになる。この場合、ベクトルの対称性を利用することにより、

・16回の演算テーブルへのアクセス

・2回の“定数 \* 変数”の演算

だけで補間点を求めることができる。

【0056】(b) 各フィルタ間で共用できる演算の保存による演算量の節約

補間点数分のフィルタを用意することによって、高速化できることはすでに述べたが、各フィルタ間で共用できる演算を保存することによって演算量を節約することができる。

(c) 4倍拡大における高速化の例

図20は、4倍拡大におけるコーディング例を示している。上述した高速化が具体的に適用されている行に注釈

を付している。すなわち、高速化は、  
 ・原点のコピー  
 ・格子線上の簡易式  
 ・各フィルタ間で共用できる演算の保存  
 ・2のn乗倍の特異点の簡易式  
 という手法で表れている。

【0057】上述してきたように整数倍の補間拡大に限れば処理量はかなり低減することができる。従って、局所的に補間位置を特定することが高速化に有効であることが分かった。かかる整数倍による補間拡大の方法を適用する場合、整数倍の補間に限定せず、少数倍の補間拡大についても応用することが考えられる。ここでは、拡大処理の他の実施例について、少数倍、具体的には1.25倍の補間拡大について考察する。この1.25倍の補間拡大における局所的な作用を図25に示す。同図において、局所的な1.25倍とは、4点の原点を補間点の1点を含めて5点にすることである。すなわち、平面において、16点の原点を補間点を含めて25点に拡大することをいう。

【0058】従って、図25に示すように補間点の座標は予め推測することが可能であり、そのことによる演算の効率化および演算量の低減は上述した整数倍の補間拡大による実施例と同様である。よって、それぞれの補間点ごとに最適なフィルタを用意すればよいことになる。ここでは、それぞれの補間点ごとのフィルタを子フィルタと呼び、16点の原点から補間点を含め25点の画素を生成する。これらの子フィルタを使用した全体のフィ

$$\cdot dL = \{(sL-1)/4\} * 5 + 1 \quad \{ \} \text{は整数部分} \\ \dots \text{式(1)}$$

となる。ただし、複数の連続した局所的な箇所であつ ※30※て、参照画素の影響がなければ、

$$\cdot dL = \{sL/4\} * 5 \quad \dots \text{式(2)}$$

となる。

【0061】次に、画像データを形成する画素数が4の倍数である場合に、1.25倍の補間拡大を実施することについて検証する。具体的に、本実施形態においてはデジタルカメラの補間拡大などにより必要とされている、「1280画素\*960画素」を1.25倍により★

$$\cdot dL\_1 = [4 * N / 4] * 5 \\ = [N] * 5 \\ = 5 * N \quad \dots \text{(3)}$$

となり、局所的な1.25倍が利用可能であることが分かる。ただし、全体の大きさを考えた場合は、式(1)★

$$\cdot dL\_2 = [4 * (N-1) / 4] * 5 + 1 \quad \dots \text{(4)} \\ = [N-1/4] * 5 + 1 \quad \dots \text{(4-1)} \\ = 5 * N - 4 \quad \dots \text{(4-2)}$$

となる。式(4-2)は、デスティネーション画像として4画素足りない状態を示し、式(4-1)は(N-1)回しか、フィルタ16to25を通していないことを示している。このことから、ソース画像でフィルタ16to25を適用することができる画素は、4\*(N-

\*ルタをフィルタ16to25と呼ぶ。ここで、フィルタ16to25を構成する子フィルタに必要なベクトル要素を示す。

・P1~P5を求めるために必要な1024倍されたベクトル要素

$$P1: \{0, 1024, 0, 0\} \\ P2: \{VA, VB, VC, VD\} \\ P3: \{VE, VF, VG, VH\} \\ P4: \{VH, VG, VF, VE\} \\ P5: \{VD, VC, VB, VA\}$$

なお、VA~VHは3次元補間関数とその補間位置から予め求められる定数である。

【0059】このように、ある画像を1.25倍にすることを考えた場合、フィルタ16to25を16点ずつ周期的に通すことになる。しかし、一般的にはフィルタ16to25を通すことができない余りの領域が出ることは自明である。従って、フィルタ16to25を適用する場合、参照画素の影響も考えなければならない。この余った領域は別のフィルタを通すことになるので、均一性がなくなり、歪みが生じる。ただし、これは人間の目の誤差の範囲内であることを前提とする。

【0060】ここで、フィルタ16to25を通した場合のソース画像とデスティネーション画像の大きさの関係を示す。ソース画像1辺の長さをsLとすると、フィルタ16to25を通して得られるデスティネーション画像の1辺の長さdLは、以下になる。

★補間拡大し、「1600画素\*1200画素」に修正するという点について検証する。かかる場合、X座標およびY座標は、Nを整数とすると、4\*Nの画素数を5\*Nの画素数に変換することを示している。式(2)を使用して、sLに4\*Nを代入すると、

★に基づいて考えなければならないので、

1)点であり、4N-4(N-1)=4点は未処理のままであることが分かる。また、デスティネーション画像としてフィルタ16to25により取り出せる画素は、5\*(N-1)であり、5N-5(N-1)=5点が生成できないことになる。従って、ソース画像の残りの4

点からデスティネーション画像における不足分5点を生成する方法を考える必要がある。

【0062】このソース画像の残りの4点からデスティネーション画像における不足分5点を生成する方法を図26示す。同図において、局所的には3点を4点にする、すなわち、 $4/3=1.3333\cdots$ 倍（以下、1.3倍）の補間拡大になる。ここで、この局所的な1.3倍の補間拡大に必要なベクトル要素は以下のようになる。

・P1～P4を求めるために必要な1024倍されたベクトル要素 10

P1: {0, 1024, 0, 0}

P2: {VK, VL, VM, VN}

P3: {VX, VY, VY, VX}

P4: {VN, VM, VL, VK}

なお、VK～VN, VX, VYは3次元補間関数とその補間位置から予め求められる定数である。

【0063】以上のことから、フィルタ16to25で処理できなかった箇所については、X方向には、フィルタ12to25X、Y方向にはフィルタ12to25Y 20  
を使用する。また、X、Y方向ともに1.25倍できない箇所はフィルタ9to16を使用する。ここで、このフィルタ12to15Yを使用する際の原点と補間点の相関を示した相関図を図27に示す（フィルタ12to25Xは90度回転した図になる。）。このように、1.25倍の補間拡大では、局所的には1.25倍と1.3倍の補間拡大を実行する。かかる場合、フィルタ16to25、フィルタ12to20X、フィルタ12to20X、フィルタ9to16を使用すると補間拡大が可能となり、これに伴い補間拡大処理の高速化を実現 30  
することが可能になる。

【0064】これらのフィルタで必要なベクトル要素をまとめると、14個のベクトル要素が必要になる。よって、それぞれのベクトル要素に0～255を掛けた、256個の要素から形成される14個の演算テーブルを使用することにより演算数を減らすことが可能になる。ここで、この演算テーブルの要素1個の大きさを4バイトとすると、1個の演算テーブルの大きさは、1024バイト（1KB）である。これが14個あるので、演算テーブルに必要なメモリは14KBとなる。

【0065】このように、ソース画像の取得処理後（手順1）、フォトインスタンス取得処理（手順2）を経たら、モジュールの側からドライバの側に向けて処理画像領域を通知するが（手順3）、このときに各画像処理モジュールで必要な周縁領域を加えていくことにより、要求された処理画像領域に必要な周縁領域を加えた上でソース画像（S-Unit）をドライバの側が登録し（手順4）、これを利用して順次画像処理（強調処理と拡大処理（手順5））を実行することになり、画像処理が周縁領域を必要とする場合にも極めてシンプルに必要領域 50

を確保して処理を実行することができる。

【図面の簡単な説明】

【図1】本発明の一実施形態にかかる画像処理プログラムを実行するコンピュータシステムのブロック図である。

【図2】印刷イメージの処理単位の変化を示す図である。

【図3】オペレーティングシステムに組み込まれた状態でのプリンタドライバの概略構成を示す図である。

【図4】画像の処理単位を示す図である。

【図5】強調処理で必要な周縁領域を示す図である。

【図6】拡大処理で必要な周縁領域を示す図である。

【図7】モジュール側からドライバ側に必要領域を通知する過程を示す図である。

【図8】S-AreaとS-Unitの関係を示す図である。

【図9】S-Unitのポインタテーブルを示す図である。

【図10】強調処理を実行する場合の実質的処理対象領域を示す図である。

【図11】拡大処理を実行する場合の実質的処理対象領域を示す図である。

【図12】プリンタドライバにおける手続の流れを示す図である。

【図13】第1のフィルタ処理による拡大処理を示す図である。

【図14】第2のフィルタ処理による拡大処理を示す図である。

【図15】二段階のフィルタ処理による拡大処理を示す図である。

【図16】3次補間の格子点と補間点の関係を示す図である。

【図17】2倍補間の格子点と補間点の関係を示す図である。

【図18】3倍補間の格子点と補間点の関係を示す図である。

【図19】4つのフィルタ処理を利用した3次補間による拡大処理を示す図である。

【図20】4倍拡大におけるコーディング例を示す図である。

【図21】S-Unitについて処理対象となるものを分離格納した場合の図である。

【図22】S-Unitの処理領域テーブルを示す図である。

【図23】他の実施の形態におけるオペレーティングシステムに組み込まれた状態でのプリンタドライバの構成を示す図である。

【図24】他の実施の形態における画像の処理単位を示す図である。

【図25】1.25倍補間の格子点と補間点の関係を示



す図である。

【図26】1. 3倍補間の格子点と補間点の関係を示す図である。

【図27】フィルタ12t o 15 Yを使用する際の原点と補間点の相関を示した相関図である。

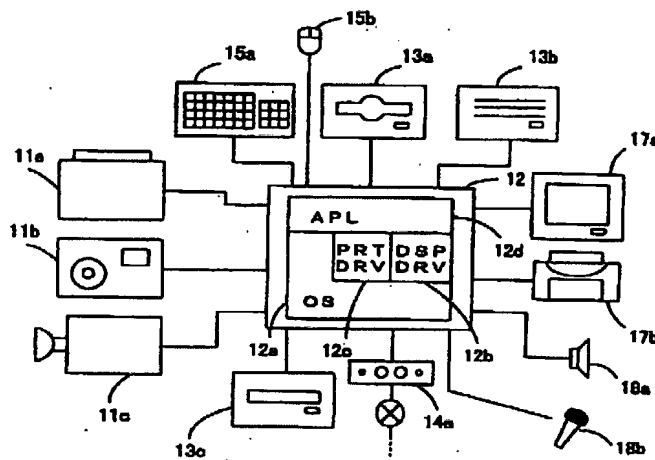
【符号の説明】

10…コンピュータシステム  
11a…スキャナ  
11a2…スキャナ  
11b…デジタルスチルカメラ  
11b1…デジタルスチルカメラ  
11b2…デジタルスチルカメラ  
11c…ビデオカメラ  
12…コンピュータ本体  
12a…オペレーティングシステム

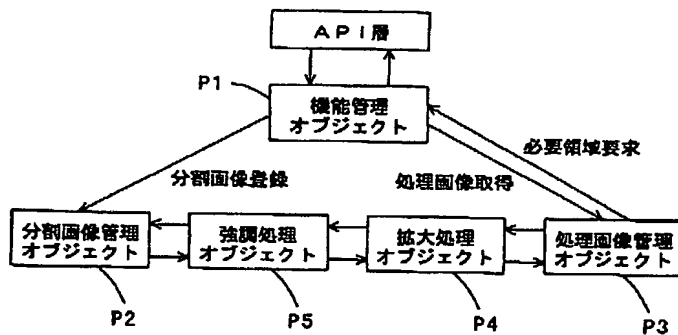
\* 12b…ディスプレイドライバ  
12c…プリンタドライバ  
12d…アプリケーション  
13a…フロッピーディスクドライブ  
13b…ハードディスク  
13c…CD-ROMドライブ  
14a…モデム  
14a2…モデム  
15a…キーボード  
15b…マウス  
17a…ディスプレイ  
17b…カラープリンタ  
18a…スピーカ  
18b…マイク

\*

【図1】

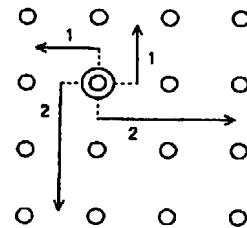


【図3】



【図5】

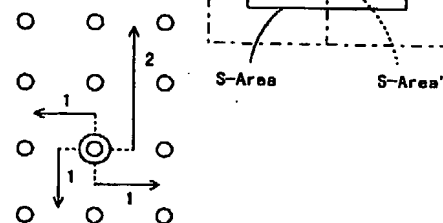
強調処理



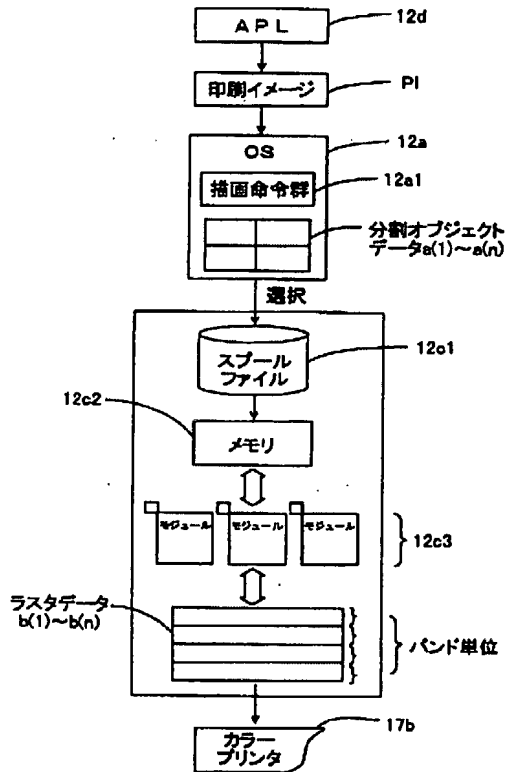
【図8】

【図6】

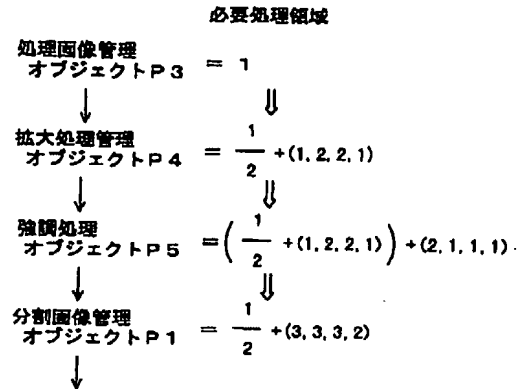
拡大処理



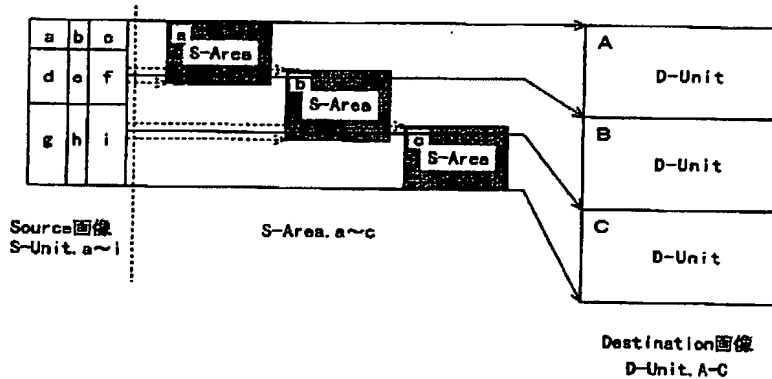
【図2】



【図7】

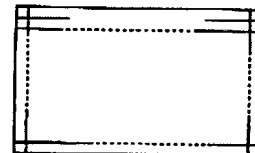


【図4】



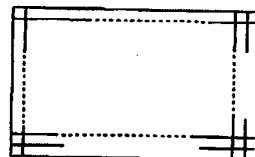
【図10】

強調処理実行

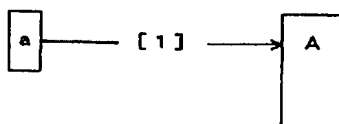


【図11】

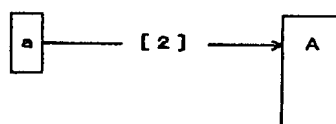
拡大処理実行



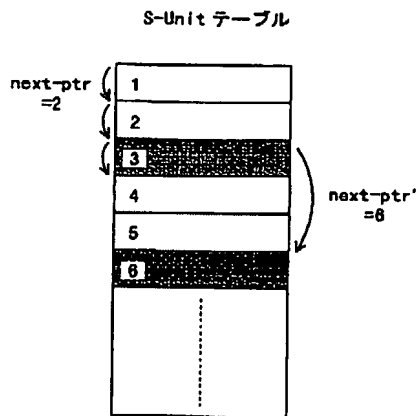
【図13】



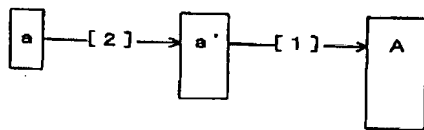
【図14】



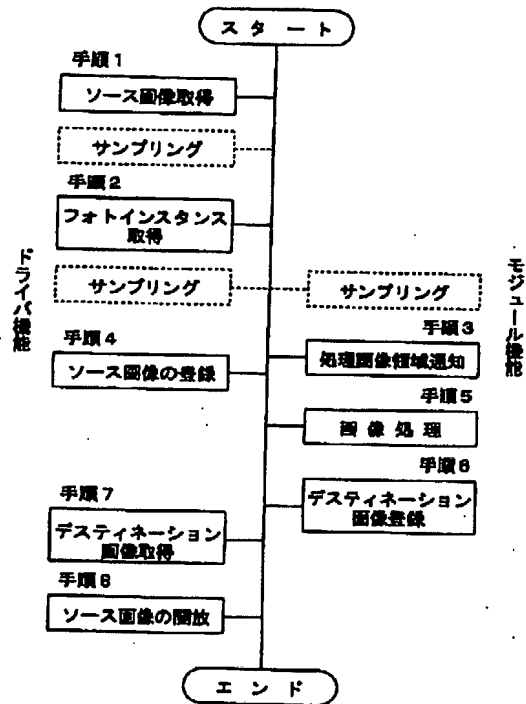
【図9】



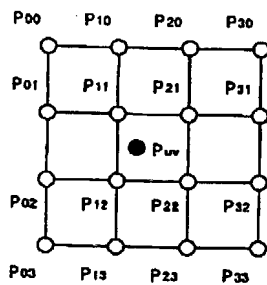
【図15】



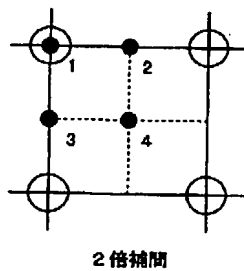
【図12】



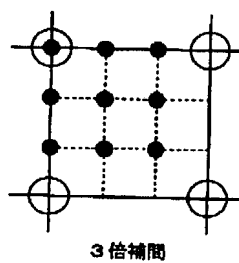
【図16】



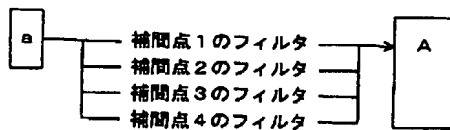
【図17】



【図18】



【図19】



【図20】

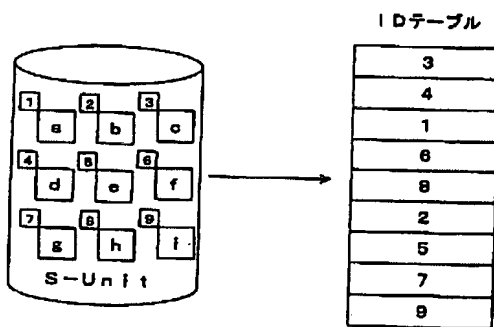
```

// ey= 0, ex=0 *****原点のコピー
dP = P11;
// ey=0, ex=1 *****格子線上の簡易式
dP = (T10[P10]+T11[P11]+T12[P12]+T13[P13]+512)>>10;
// ey=0, ex=2 *****格子線上の簡易式
dP = (T20[P10]+T21[P11]+T22[P12]+T23[P13]+512)>>10;
// ey=0, ex=3 *****格子線上の簡易式
dP = (T30[P10]+T31[P11]+T32[P12]+T33[P13]+512)>>10;

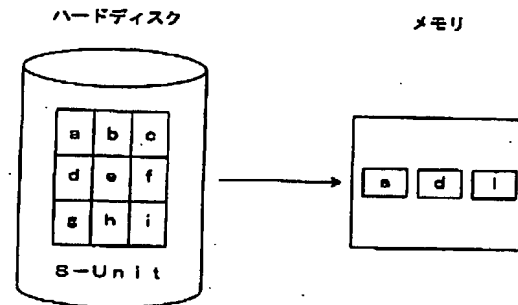
for( ey = 1; ey < 4; ey++ ) {
  各フィルタ間で共用できる演算の保存
  switch( ey ) {
    case 1:
      for( i = 0; i < 4; i++ ) {
        Px[i] = (T10[P0i]+T11[P1i]+T12[P2i]+T13[P3i]);
      } break;
    case 2:
      for( i = 0; i < 4; i++ ) {
        Px[i] = (T20[P0i]+T21[P1i]+T22[P2i]+T23[P3i]);
      } break;
    case 3:
      for( i = 0; i < 4; i++ ) {
        Px[i] = (T30[P0i]+T31[P1i]+T32[P2i]+T33[P3i]);
      } break;
  }
  // ey=n, ex=0 *****格子線上の簡易式
  dP = (Px[1]+512) >> 10;
  // ey=n, ex=1 *****
  dP = (Px[0]*M40+Px[1]*M41+Px[2]*M42+
        Px[3]*M43+524288) >> 20;
  // ey=n, ex=2 *****2のn乗倍の特異点の簡易式
  dP = (Px[0]+Px[3]*M44+Px[1]*M45
        +524288) >> 20;
  // ey=n, ex=3 *****
  dP = (Px[0]*M43+Px[1]*M42+Px[2]*M41+
        Px[3]*M40+524288) >> 20;
}

```

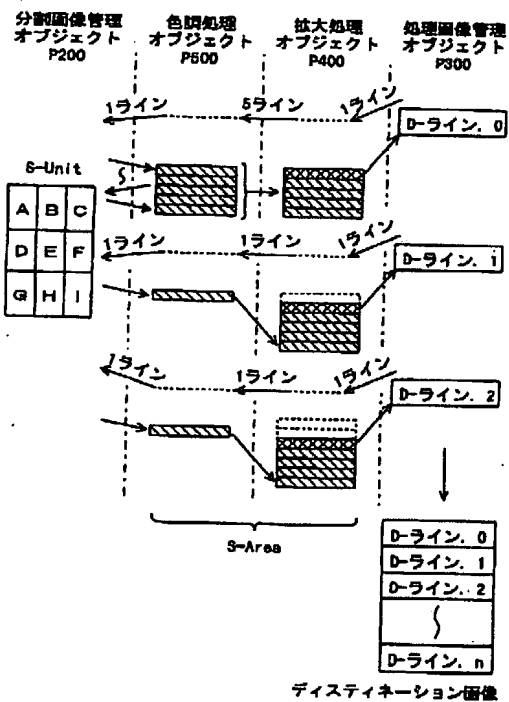
【図22】



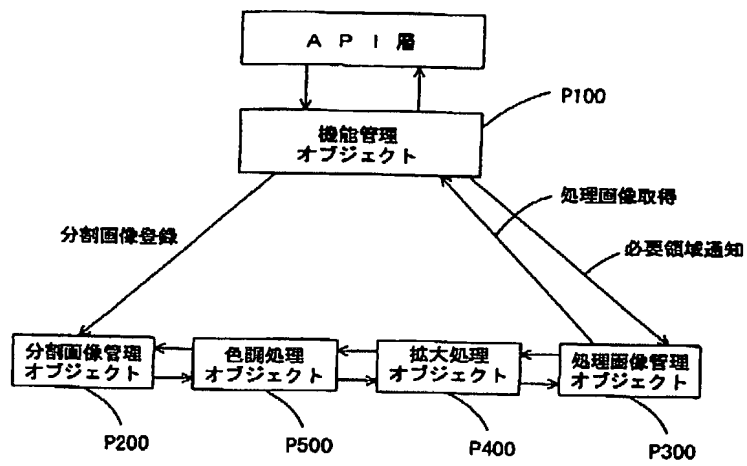
【図21】



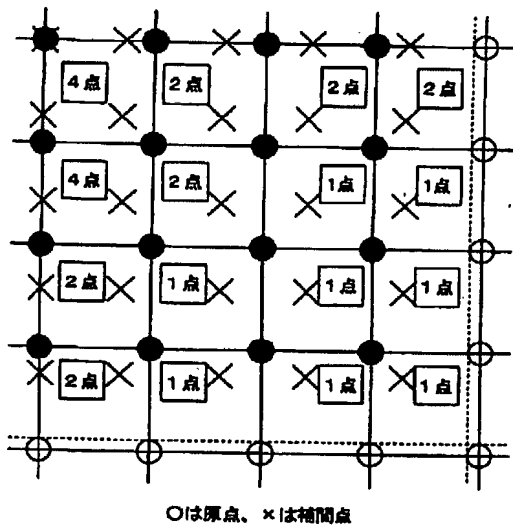
【図24】



【図23】



【図25】



【図26】



【図27】

